Programming Fundamentals
Lecture #13
Arrays

Junaid Hassan

Lecturer CS & IT Department UOS MBDIN

junaidte14@gmail.com

Computing Mean, Median and Mode using Arrays:

Let's say that we have an array of 55 values, then:

The mean is the arithmetic average of the 55 values. Function mean (will develop later) computes the mean by totaling the 55 elements and dividing the result by 55

The median is the "middle value." Function median (will develop later) determines the median by calling function bubbleSort (function) to sort the array of responses into ascending order, then picking the middle element, SIZE / 2, of the sorted array

The mode is the value that occurs most frequently among the 55 responses. Function mode (will develop later) determines the mode by counting the number of responses of each type, then selecting the value with the greatest count

```
#include <stdio.h>
#define SIZE 99

/* function prototypes */
void mean( const int answer[] );
void median( int answer[] );
void mode( int freq[], const int answer[] ) ;
void bubbleSort( int a[] );
void printArray( const int a[] );
```

```
int frequency[ 10 ] = { 0 }; /* initialize array frequency */

/* initialize array response */
int response[ SIZE ] =
   { 6, 7, 8, 9, 8, 7, 8, 9, 8, 9,
     7, 8, 9, 5, 9, 8, 7, 8, 7, 8,
     6, 7, 8, 9, 3, 9, 8, 7, 8, 7,
     7, 8, 9, 8, 9, 8, 9, 7, 8, 9,
     6, 7, 8, 7, 8, 7, 9, 8, 9, 2,
     7, 8, 9, 8, 9, 8, 9, 7, 5, 3,
     5, 6, 7, 2, 5, 3, 9, 4, 6, 4,
     7, 8, 9, 6, 8, 7, 8, 9, 7, 8,
     7, 4, 4, 2, 5, 3, 8, 7, 5, 6,
     4, 5, 6, 1, 6, 5, 7, 8, 7 };

/* process responses */
mean( response );
median( response );
mode( frequency, response );
return 0; /* indicates successful termination */
```

```c
/* calculate average of all response values */
void mean( const int answer[] )
{
   int j; /* counter for totaling array elements */
   int total = 0; /* variable to hold sum of array elements */

   printf( "%s\n%s\n%s\n", "********", "  Mean", "********" );

   /* total response values */
   for ( j = 0; j < SIZE; j++ ) {
      total += answer[ j ];
   } /* end for */

   printf( "The mean is the average value of the data\n"
           "items. The mean is equal to the total of\n"
           "all the data items divided by the number\n"
           "of data items ( %d ). The mean value for\n"
           "this run is: %d / %d = %.4f\n\n",
           SIZE, total, SIZE, ( double ) total / SIZE );
} /* end function mean */
```

```c
/* sort array and determine median element's value */
void median( int answer[] )
{
   printf( "\n%s\n%s\n%s\n%s",
           "********", " Median", "********",
           "The unsorted array of responses is" );

   printArray( answer ); /* output unsorted array */

   bubbleSort( answer ); /* sort array */

   printf( "\n\nThe sorted array is" );
   printArray( answer ); /* output sorted array */

   /* display median element */
   printf( "\n\nThe median is element %d of\n"
           "the sorted %d element array.\n"
           "For this run the median is %d\n\n",
           SIZE / 2, SIZE, answer[ SIZE / 2 ] );
} /* end function median */
```

```c
/* determine most frequent response */
void mode( int freq[], const int answer[] )
{
   int rating; /* counter for accessing elements 1-9 of array freq */
   int j; /* counter for summarizing elements 0-98 of array answer */
   int h; /* counter for diplaying histograms of elements in array freq */
   int largest = 0; /* represents largest frequency */
   int modeValue = 0; /* represents most frequent response */

   printf( "\n%s\n%s\n%s\n",
           "********", "  Mode", "******** );
```

```c
/* initialize frequencies to 0 */
for ( rating = 1; rating <= 9; rating++ ) {
   freq[ rating ] = 0;
} /* end for */

/* summarize frequencies */
for ( j = 0; j < SIZE; j++ ) {
   ++freq[ answer[ j ] ];
} /* end for */

/* output headers for result columns */
printf( "%s%11s%19s\n\n%54s\n%54s\n\n",
        "Response", "Frequency", "Histogram",
        "1    1    2    2", "5    0    5    0    5" );
```

```
   /* output results */
   for ( rating = 1; rating <= 9; rating++ ) {
      printf( "%8d%11d          ", rating, freq[ rating ] );

      /* keep track of mode value and largest frequency value */
      if ( freq[ rating ] > largest ) {
         largest = freq[ rating ];
         modeValue = rating;
      } /* end if */

      /* output histogram bar representing frequency value */
      for ( h = 1; h <= freq[ rating ]; h++ ) {
         printf( "*" );
      } /* end inner for */

      printf( "\n" ); /* being new line of output */
   } /* end outer for */

   /* display the mode value */
   printf( "The mode is the most frequent value.\n"
           "For this run the mode is %d which occurred"
           " %d times.\n", modeValue, largest );
} /* end function mode */
```

```
/* output array contents (20 values per row) */
void printArray( const int a[] )
{
   int j; /* counter */

   /* output array contents */
   for ( j = 0; j < SIZE; j++ ) {

      if ( j % 20 == 0 ) { /* begin new line every 20 values */
         printf( "\n" );
      } /* end if */

      printf( "%2d", a[ j ] );
   } /* end for */
} /* end function printArray */
```

Searching Arrays:

The process of finding a particular element of an array is called searching

```c
/* compare key to every element of array until the location is found
   or until the end of array is reached; return subscript of element
   if key or -1 if key is not found */
int linearSearch( const int array[], int key, int size )
{
   int n; /* counter */

   /* loop through array */
   for ( n = 0; n < size; ++n ) {

      if ( array[ n ] == key ) {
         return n; /* return location of key */
      } /* end if */
   } /* end for */

   return -1; /* key not found */
} /* end function linearSearch */
```

Multiple Sub-scripted Arrays:

Arrays in C can have multiple subscripts. A common use of multiple-subscripted arrays (also called multidimensional arrays) is to represent tables of values consisting of information arranged in rows and columns. To identify a particular table element, we must specify two subscripts: The first (by convention) identifies the element's row and the second (by convention) identifies the element's column.

Tables or arrays that require two subscripts to identify a particular element are called double-subscripted arrays.

int b[2][2] = { { 1, 2 }, { 3, 4 } };

To get 3 from above array:

we will use: b[1][0]